IN THE CLAIMS:

The text of all pending claims, (including withdrawn claims) is set forth below. Cancelled and not entered claims are indicated with claim number and status only. The claims as listed below show added text with underlining and deleted text with ~~strikethrough~~. The status of each claim is indicated with one of (original), (currently amended), (cancelled), (withdrawn), (new), (previously presented), or (not entered).

1. (currently amended) A volatile or non-volatile computer-readable media or storage unit storing a data structure for an original mesh surface that is provided with a bounding surface bounding the mesh surface and is provided with a tight inner surface, the data structure comprising:

a first tessellation linking the tight inner surface to the original mesh surface; ~~and~~

a second tessellation linking the bounding surface to the tight inner surface; and

a program used for displaying the original mesh on a display by a computer using the first and second tessellations.

2. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 1, wherein the tight inner surface comprises a convex hull surface.

3. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 1, wherein the second tessellation is at least partially separate from the bounding surface and the tight inner surface and is constrained by vertices of those surfaces, and wherein the first tessellation is at least partially separate from the original surface and the convex hull surface and is constrained by vertices of those surfaces.

4. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 3, wherein the tight inner surface comprises a convex hull surface.

5. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 3, wherein the tessellations comprise cell-like structures of interrelated polygons or polyhedrons of two or more dimensions.

6. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 1, wherein the tight inner surface and the original mesh surface have at least some common vertices.

7. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 1, wherein the bounding surface is sufficiently simple to enable a trivial determination of whether it is intersected.

8. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 1, wherein the second tessellation tessellates a space between the bounding surface and the tight inner surface, and wherein the second tessellation tessellates a space between the tight inner surface and the original surface mesh.

9. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 8, wherein the tight inner surface is a convex hull surface.

10. (original) A volatile or non-volatile computer-readable media or storage unit according to claim 1, wherein the bounding surface has a low resolution relative to the original surface mesh.

11. (currently amended) A method of using a volatile or non-volatile computer-readable media or storage unit ~~according to any of claims 1 to 10~~, the method comprising:

storing a data structure for an original mesh surface that is provided with a bounding surface bounding the mesh surface and is provided with a tight inner surface, the data structure comprising:

a first tessellation linking the tight inner surface to the original mesh surface; and

a second tessellation linking the bounding surface to the tight inner surface, and

using the data structure to at least one of: find an intersection with the original mesh surface; determine whether an intersection with the original mesh surface is occluded by the original mesh surface; and to identify an order of an intersection.

12. (original) A method according to claim 11, wherein the order is an order of the intersection relative to other intersections.

13. (original) A method according to claim 11, wherein the identifying the order further comprises for any given N identifying an Nth closest or Nth outermost intersection or identifying N outermost or closest intersections.

14. (original) A method according to claim 11, wherein the finding, determining, or identifying is also based on a line intersecting the original mesh surface.

15. (original) A method according to claim 11, wherein the finding comprises finding an intersection with the bounding surface, and using that intersection to find the intersection with the original mesh surface.

16. (original) A method according to claim 15, further comprising traversing a path from the intersection with the bounding surface through the second tessellation to an intersection with the convex hull surface, and traversing through the first tessellation to thereby find the intersection with the original mesh surface.

17. (original) A method according to claim 16, wherein the path corresponds to polygons or polyhedrons of the tessellations that intersect with a line that also determines the intersection with the bounding surface and also determines the intersection with the original mesh surface.

18. (previously presented) A method of automatically finding an intersection with an original mesh surface, the method comprising:

finding an intersection with the original mesh surface by using an outer bounding surface, a tight inner surface that is both bounded by the outer bounding surface and wraps the original mesh surface and a tessellation between the inner bounding surface and the outer bounding surface; and

one of displaying the intersection on a display and storing the intersection in a storage.

19. (previously presented) A method of automatically finding an intersection with an original mesh surface, the method comprising

finding an intersection with the original mesh surface by using an outer bounding surface, a tight inner surface that is both bounded by the outer bounding surface and wraps the original mesh surface and a tessellation between the inner bounding surface and the outer bounding

surface, wherein the finding of the intersection with the original mesh surface is performed according to an intersection with the outer bounding surface; and

one of displaying the intersection on a display and storing the intersection in a storage.

20. (previously presented) A method of automatically finding an intersection with an original mesh surface, the method comprising:

finding an intersection with the original mesh surface by using an outer bounding surface and a tight inner surface that is both bounded by the outer bounding surface and wraps the original mesh surface, wherein the finding of the intersection with the original mesh surface is performed according to an intersection with the outer bounding surface, and wherein the tight inner surface comprises a convex hull, wherein there is a first tessellation between and linking the convex hull with the original mesh surface, wherein there is a second tessellation between and linking the convex hull with the outer bounding surface, and wherein the finding further comprises traversing a path from the intersection with the outer bounding surface through the second tessellation to an intersection with the convex hull surface, and traversing from the intersection with the convex hull surface through the first tessellation to thereby find the intersection with the original mesh surface; and

one of displaying the intersection on a display and storing the intersection in a storage.

21. (cancelled)

22. (previously presented) A method of automatically finding an intersection with an original mesh surface, the method comprising:

finding an intersection with the original mesh surface by using an outer bounding surface and a tight inner surface that is both bounded by the outer bounding surface and wraps the original mesh surface, wherein the finding of the intersection with the original mesh surface is performed according to an intersection with the outer bounding surface, and wherein the tight inner surface comprises a convex hull, wherein there is a first tessellation between and linking the convex hull with the original mesh surface, wherein there is a second tessellation between and linking the convex hull with the outer bounding surface, and wherein the finding further comprises traversing a path from the intersection with the outer bounding surface through the second tessellation to an intersection with the convex hull surface, and traversing from the intersection with the convex hull surface through the first tessellation to thereby find the intersection with the original mesh surface, wherein the path corresponds to polygons or

polyhedrons of the tessellations that intersect with a line that also defines the intersection with the bounding surface and also defines the intersection with the original mesh surface; and

one of displaying the intersection on a display and storing the intersection in a storage.

23. (previously presented) A method of intersection determining where bounding polyhedrons or polygons bound and are tightly constrained to a mesh model with a tessellation, the method comprising:

using the polyhedrons or polygons tessellation to automatically determine an intersection order or rank of one or more intersections between a line and the mesh model; and

one of displaying the intersection on a display and storing the intersection in a storage.

24. (currently amended) A method of automatically determining whether an intersection between a line and a mesh model is, relative to the mesh model, an outermost intersection between the line and the mesh model, the method comprising:

determining whether a polyhedron or polygon intersected by the line one of contains ~~(or is)~~and is an outermost intersection with the line based on whether such polyhedron or polygon is on a convex hull surface of the mesh model using a tessellation linking the mesh model and the convex hull; and

one of displaying the intersection on a display and storing the intersection in a storage.

25. (currently amended) A method of automatically determining whether an intersection between a line and a mesh model is, relative to the mesh model, an outermost intersection between the line and the mesh model, the method comprising:

determining whether a polyhedron or polygon intersected by the line one of contains ~~(or is)~~ and is an outermost intersection with the line based on whether such polyhedron or polygon is on a convex hull surface of the mesh model using a tessellation linking the mesh model and the convex hull, further comprising identifying the polyhedron or polygon by traversing polyhedrons or polygons that intersect the line; and

one of displaying the intersection on a display and storing the intersection in a storage.

26. (currently amended) A method of automatically determining whether an intersection between a line and a mesh model is, relative to the mesh model, an outermost intersection between the line and the mesh model, the method comprising:

determining whether a polyhedron or polygon intersected by the line one of contains ~~(or is)~~ and is an outermost intersection with the line based on whether such polyhedron or polygon is on a convex hull surface of the mesh model using a tessellation linking the mesh model and the convex hull, further comprising traversing to a next polyhedron or polygon when a traversed polyhedron or polygon is inside an interior or convex region of the mesh model; and

one of displaying the intersection on a display and storing the intersection in a storage.

27. (currently amended) A method of determining a ray-object intersection, where there has been a first intersection between a ray and a mesh object, and where the ray or the mesh object move relative to the other, the method comprising:

finding, based on the first intersection, a second intersection between the ray and the mesh object based on the moving of the ray or the mesh object; and

after the finding, determining whether the second intersection is occluded along the ray by the mesh object using a tessellation constrained to the mesh model; and

one of displaying the intersection on a display and storing the intersection in a storage.

28. (currently amended) A method of determining a ray-object intersection, where there has been a first intersection between a ray and a mesh object, and where the ray or the mesh object move relative to the other, the method comprising:

finding, based on the first intersection, a second intersection between the ray and the mesh object based on the moving of the ray or the mesh object; and

after the finding, determining whether the second intersection is occluded along the ray by the mesh object using a tessellation constrained to the mesh model, wherein the second intersection is coherent or locally near the first intersection; and

one of displaying the intersection on a display and storing the intersection in a storage.

29. (currently amended) A method of determining a ray-object intersection, where there has been a first intersection between a ray and a mesh object, and where the ray or the mesh object move relative to the other, the method comprising:

finding, based on the first intersection, a second intersection between the ray and the mesh object based on the moving of the ray or the mesh object; and

after the finding, determining whether the second intersection is occluded along the ray by the mesh object using a tessellation constrained to the mesh model, further comprising finding a third intersection that occludes along ray; and

one of displaying the intersection on a display and storing the intersection in a storage.

30. (currently amended) A method of determining a ray-object intersection, where there has been a first intersection between a ray and a mesh object, and where the ray or the mesh object move relative to the other, the method comprising:

finding, based on the first intersection, a second intersection between the ray and the mesh object based on the moving of the ray or the mesh object;

after the finding, determining whether the second intersection is occluded along the ray by the mesh object;

finding a third intersection that occludes along the ray, and wherein the third intersection is found by traversing polygons or polyhedrons of a tessellation constrained to the mesh model; and

one of displaying the third intersection on a display and storing the intersection in a storage.

31. (previously presented) A method of repeatedly moving an intersection of a ray with a mesh object, the method comprising:

detecting movements of the ray or the object, one relative to the other, and for some of the movements:

when the ray intersects the mesh object at a local neighbor of a face of the mesh object, determining whether intersection of the ray with the mesh object is occluded by the mesh object by traversing polygons not part of the mesh object; and

when the ray does not intersect the mesh object at a local neighbor of a face of the mesh object, finding an intersection of the ray with the mesh object by traversing polygons intersected by the ray, where the polygons are not part of the mesh object and include at least one polygon of a bounding surface bounding the mesh object; and

one of displaying the moved intersection on a display and storing the moved intersection in a storage.

32. (currently amended) A volatile or non-volatile computer-readable storage storing a data structure constrained to a complex high-resolution original mesh surface for controlling a computer, the data structure comprising;

a bounding mesh bounding the original surface mesh and with low resolution relative to the original mesh surface; ~~and~~

a tessellation mesh of polyhedrons constrained to the original mesh surface and constrained to the bounding mesh, where polyhedrons with shared faces generally decrease in size in the direction of the original mesh surface; and

a program used for displaying the original mesh surface on a display by a computer using the bounding mesh and the tessellation mesh.

33. (original) A volatile or non-volatile computer-readable media or storage unit storing a data structure according to claim 32, wherein the data structure is further constrained to a bounding surface of the original mesh surface.

34. (original) An apparatus, comprising:

a storage unit storing: an original mesh surface that is provided with a bounding surface bounding the mesh surface, a convex hull surface of the original mesh surface, a first tessellation linking the convex hull to the original mesh surface, and a second tessellation linking the bounding surface to the convex hull, where the second tessellation tessellates a space between the bounding surface and the convex hull surface, and where the first tessellation tessellates a space between the convex hull surface and the original surface mesh; and

a processing unit performing at least one of:

finding a first intersection between a ray and the original mesh by finding a first intersected polygon or polyhedron of the bounding surface, and then traversing adjacent intersected polygons or polyhedrons starting from the first intersection until the intersection is found; and

finding a second intersection between the ray and the original mesh when the ray or original mesh have relatively moved, finding a polygon locally neighboring the first intersection and containing a first intersection with the moved ray, and traversing out from the neighbor polygon through adjacent polygons or polyhedrons intersected by the moved ray, and determining whether traversed polygons or polygons of traversed polyhedrons are unoccluded along ray based on whether they are part of the convex hull surface.

35. (previously presented) A method of automatically finding an intersection with an original mesh surface, the method comprising:

finding an intersection with the original mesh surface by using an outer bounding surface, using a tight inner surface that is both bounded by the outer bounding surface and wraps the original mesh surface and using details of an intersection with the outer bounding surface; and

one of displaying the intersection on a display and storing the intersection in a storage.

36. (previously presented) A method of automatically finding an intersection with an original mesh surface, the method comprising:

finding an intersection with the original mesh surface by using an outer bounding surface and a tessellation between the outer bounding surface and the original mesh surface; and

one of displaying the intersection on a display and storing the intersection in a storage.